VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Informační systém pro řízení projektů

Information System for Projects Management

2015 Filip Moták

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Filip Moták**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Informační systém pro řízení projektů
Information System for Projects Management

Zásady pro vypracování:

Hlavním cílem bakalářské práce bude vytvořit informační systém, který bude pomáhat při řízení projektů. Primární scénář použití projektu lze shrnout v těchto bodech:
1. Systém umožní zakládat projekty. Projekt bude složen z různých úkolů, událostí či milníků.
2. K projektu budou přiřazeni uživatelé, kteří se podílejí na jeho řešení a IS umožní rozdělit podíl či zodpovědnost za řešení jednotlivých úloh.
3. IS poskytne různé statistiky či grafy, které zobrazí aktuální stav řešení celého projektu, či plnění přiřazených úkolů.

IS se bude skládat ze serverové části a části klientské. Pro řešení použijte technologie platformy .NET a programovací jazyk C#. Klientská aplikace bude realizována pro mobilní zařízení využívající Windows Phone 8 a bude umožňovat jak online, tak offline provoz.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marek Běhálek, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2015 .....

Na tomto místě bych rád poděkoval všem, kteří mi pomohli k úspěšnému dokončení této práce, zejména mému vedoucímu panu Ing. Marku Běhálkovi, Ph.D. za cenné rady a podporu.

Abstrakt

Bakalářská práce Informační systém pro řízení projektů se zabývá vytvořením systému pro správu projektů. Tento systém je určen hlavně pro menší firmy a domácnosti, spolky a kluby. Práce umožní pracovníkům získat přehled nad řešením projektu a také kontrolu nad jednotlivými úkoly. Výsledný systém se skládá z klientské aplikace pro Windows Phone 8.1 a ze serverové aplikace nad architekturou .NET. Bakalářská práce také rozebírá celou tématiku po teoretické stránce od analýzy uvodních požadavků přes návrh až po výsledné řešení.

Klíčová slova: Informační systém, IS, Projekty, Řízení projektů

Abstract

This bachelor thesis Information System for Project Management focuses on development of information system for managing simple to semi-simple projects. The system is mainly designed for small companies, families or clubs. System provides workers with the ability to control the flow of the project and view overall progress of project fulfillment. Final product consists of client application for Windows Phone 8.1 and server application base on architecture .NET. The bachelor thesis also studies the whole subject from the theoretical point of view, from the analysis of the original requirements, through design, to the final solution.

Keywords: Information system, IS, Projects, Project management

Seznam použitých zkratek a symbolů

IS	–	Informační systém
WCF	–	Windows Community Foundation
EF	–	Entity Framework
XAML	–	Extensible Application Markup Language
WPF	–	Windows Presentation Foundation
UI	–	Uživatelské rozhranní
VB	–	Visual Basic
JSON	–	JavaScript Object Notation
XML	–	Extensible Markup Language
EAP	–	Event-based Asynchronous Pattern
WP	–	Windows Phone
BTA	–	Background Task Agent

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Hlavním cílem bakalářské práce je navrhnout systém, který umožní malým společnostem, domácnostem, klubům či spolkům spravovat a kontrolovat skupinové projekty. Systém umožní uživateli založit projekt, pozvat k projektu spolupracovníky a řadit je do skupin. K projektu může uživatel přidat jednotlivé milníky, události či příspěvky. Jednotlivé milníky se skládají z dílčích úkolů a pod-úkolů, úkoly na sobě mohou být závislé - nelze pracovat na jednom úkolu, pokud předtím nebyl splněn úkol druhý. Uživatel může k úkolům přiřazovat členy, tím jim přiřadit zodpovědnost plnění úkolu.

Druhá část bakalářské práce se zabývá vývojem klientské aplikace, která bude pracovat s IS. Ta je řešena jako mobilní aplikace pro platformu Windows Phone 8. Mobilní klient umožňuje offline provoz a z přijatých dat je schopen sestavit statistiky řešení projektu.

2 Existující aplikace pro správu projektů

Před samotným řešením práce jsem provedl krátký průzkum podobných systémů, řešících stejný problém. V dalších kapitolách se pokusím popsat jejich rysy a funkce a porovnat je s předmětem této práce.

Na internetu lze nalézt mnoho systému, řešící stejnou problematiku. Tyto systémy se liší hlavně přístupem k rozdělení privilegií a struktuře projektů. Podporované platformy jsou také považovány za důležitý faktor. Následuje výčet alternativních systémů pro správu projektů.

2.1 Basecamp

Basecamp [2] je webová služba pro správu projektu, která umožňuje rozdělit kompetence pracovníků projektu, vytvářet to-do listy, nahrávat dokumenty a soubory a diskutovat o jednotlivých částech projektu. Systém umožňuje vedle základního zobrazení informací také časové rozložení dat ve formě kalendáře.

Služba si zakládá na rozdělení privilegií pracovníků, nadřízených a klientů. Basecamp je přístupný přes internetový prohlížeč a na mobilních zařízeních se systémem iOS a Android.

2.2 Copper

Copper [13] je další webová služba pro projektový management. Stejně jako Basecamp umožňuje vytváření projektů a rozdělení kompetencí uživatelů, Copper ale toto rozdělení rozšiřuje o vytváření organizací. Všechny úkoly projektu jsou řazeny na časové ose, kde každý úkol má daný čas počátku a odhadovaný čas řešení. Copper je dostupný pouze z internetového prohlížeče.

2.3 Huddle

Dalším reprezentantem software pro řízení projektů je webová aplikace Huddle [7]. Tato aplikace se prezentuje hlavně jako bezpečná služba pro sdílení, synchronizaci a správu souborů s propracovaným systémem pro spolupráci interních a externích uživatelů.

Funkce tohoto systému jsou víceméně stejné jako u již zmíněných aplikací. Huddle poskytuje své služby pro Android, iOS zařízení a webové prohlížeče.

2.4 ZOHO Projects

Zoho corporation poskytuje mnoho služeb pro podnikání a jednou z nich je i aplikace pro správu projektů — ZOHO Projects [4]. Aplikace umožňuje dělit úkoly na milníky, úkoly

a podúkoly a tím vytvářet srozumitelnou strukturu, připravenou pro řešení. Stejně jako Basecamp a Huddle poskytuje tento systém možnost spolupráce zaměstnanců, klientů a dalších osob.

Tato aplikace navíc poskytuje vizualizaci dat pomocí grafů a statistik a umožňuje zobrazit vytíženost jednotlivých řešitelů projektu. Specialitou tohoto systému je tzv. „Ganttův diagram“, druh pruhového diagramu zachycující řešení projektů v čase zobrazující také míru jeho dokončení. Tento diagram není použit jen pro zobrazování dat, ale je plně interaktivní, lze v něm tedy přidávat a upravovat jednotlivé části, zobrazovat detaily milníků a úkolu.

ZOHO Projects poskytuje webové rozhraní a aplikace pro Android a iPhone.

2.5 Nozbe

Systém Nozbe [12] se zaměřuje na implementaci široce využívaných aplikací, jako například Google Calendar, Dropbox a Evernote. U úkolů v tomto systému lze nastavit množství parametrů, jako například priorita úkolů, jeho kategorie, či doba začátku a trvání. Úkol jde také nastavit jako opakující se, je tedy znovu otevírán periodicky. Nozbe neposkytuje hlubší strukturu úkolů, úkoly je maximálně možno rozdělit do kategorií.

Tuto aplikaci lze nainstalovat na zařízení se systémem Windows, Linux a Mac zařízení, Android a iPhone telefony a je přístupná přes webové rozhraní.

2.6 Shrnutí

Po zanalyzování vhodných funkcí všech systémů a jejich nedostatků jsem zvolil následující rysy, které by měla má aplikace splňovat:

- Jednoduchá registrace;
- Rychlé intuitivní vytvoření projektů a úkolů;
- Možnost rozdělení kompetencí v rámci projektu;
- Rozdělení struktury projektu na milníky, úkoly a pod-úkoly;
- Vizualizace informací o projektech pomocí grafů;
- Offline funkčnost.

3 Použité technologie

Pro řešení problému jsem využil řadu technologií, které jsou popsány v následujících sekcích.

3.1 Serverová část

Serverová část systému využívá řadu technologií, následující sekce popíší každou z nich.

3.1.1 Microsoft SQL server

Pro uložení dat se na serverové části jsem zvolil databázový systém Microsoft SQL Server, zejména kvůli výborné kompatibilitě s „dotnetovskou“ technologií.

3.1.2 Entity framework - Code first

O objektově relační mapování se stará Entity framework. Tento framework zásadně zjednodušuje práci s databází, sleduje změny jednotlivých entit a zajišťuje automatické migrace databáze při změně modelu. Code First umožňuje vytvoření databázového modelu z doménových objektů. Entity framework dále dokáže sledovat vazby mezi objekty a tyto pak mapovat do pevného úložiště.

3.1.3 Windows Community Foundation

Klient přistupuje k serveru přes servisní vrstvu, kterou zajišťuje WCF. Windows Community Foundation je schopen přenášet data přes HTTP, HTTPS a TCP, jak šifrované, tak nešifrované. Mezi další rysy WCF patří možnost zasílat data zabezpečených kanálem či pomocí zpráv. Jakožto technologie postavená na architektuře .NET, WCF poskytuje nespočetné množství metod pro využití v jazycích C# a Visual Basic, které využívám na serverové i klientské části.

3.2 Klientská část

Také klientská část využívá různé technologie pro přístup a ukládání do databáze a pro zobrazování uživatelského rozhraní.

3.2.1 SQL Server Compact

Windows Phone poskytuje pro lokální uložení dat pouze jednu databázi a tou je SQL Server Compact. S programem komunikuje přes proxy, kterou je tzv. Context. Pro dotazy nad databází se používá technologie LINQ to SQL

3.2.2 LINQ to SQL

LINQ to SQL poskytuje objektově relační mapování pro SQL Server Compact databázi. Jak již z názvu vyplývá, funguje na principu LINQ dotazu komunikujícím s relační (T-SQL orientovanou) databází. Stejně jako EF, i LINQ to SQL umožnuje mapovat doménové objekty na tabulky databáze.

3.2.3 Extensible Application Markup Language

XAML je hlavní značkový jazyk využívaný pro definici uživatelského rozhraní ve WPF, Silverlight a Windows Phone aplikacích. I když je veškerý kód pro UI možno napsat kódem v C# nebo VB, XAML poskytuje jednoduchou a pohodlnou alternativu například pomocí možnosti vázaní vlastností objektů k jednotlivým atributům grafických prvků.

3.3 Použité knihovny

Při řešení práce jsem využil řadu knihoven, které mi ulehčily práci se serializací dat do JSON formátu, dále knihovny rozšiřující již existující sadu komponent a JavaScriptovou knihovnu pro grafy. Tyto jsou popsány v následujících sekcích.

3.3.1 Json.NET

Json.NET [10] společnosti Newtonsoft je nejpopulárnější knihovna pro serializaci dat do formátu JSON nad technologií .NET. Tato knihovna poskytuje serializaci i deserializaci dat, převod mezi XML a JSON a vysoký výkon, přičemž si zachovává svou jednoduchost a pro open source software je poskytován zdarma.

3.3.2 Windows Phone Toolkit

Tato knihovna rozšiřuje základní komponenty Windows Phone, které jsou pro intuitivní použití aplikace nedostačující. Windows Phone Toolkit [9] přináší komponenty jako ListBox, LongListMultiSelector, Context menu a přepínací tlačítko Switch.

3.3.3 Highcharts

Pro účely vizualizace dat využívám JavaScriptovou knihovnu Highcharts [1]. Tato knihovna dokáže vytvářet grafy, které jsou jak informativní, tak uživatelsky a vizuálně přívětivé.

4 Specifikace požadavků

Všechny aplikace zmíněné v sekci 2 se snaží poskytnout co nejjednodušší možnost registrace do systému. Tyto aplikace jsou většinou placeny formou měsíčních poplatků, ale poskytují zkušební dobu kolem jednoho měsíce. Můj systém nebyl vytvořen se záměrem zisku, poskytuje tedy registraci bez nutnosti zadání jakýchkoliv údajů o účtu či platební kartě.

Ne všechny aplikace, které jsem zkoušel měly intuitivní ovládání. Například systém Nozbe obsahuje obrovské množství funkcí, ale pro začínajícího uživatele může být jeho grafické rozhraní matoucí. Snažil jsem se grafické rozhraní klienta co nejvíce zjednodušit a přizpůsobit jeho používání i méně technicky znalým uživatelům.

Ve struktuře projektů se systémy už rozcházejí, některé dělí projekt na dílčí části, jiné dávají více prostoru k řešení a poskytují tedy uživateli větší volnost. Pro vlastní systém jsem zvolil první variantu, systém je určen pro menší spolky a sdružení, které využijí hlavně možnost milníků s úkoly. Tento striktnější systém by měl zajistit korektní plnění projektu a jeho jednotlivých částí. Rozdělení projektu na dílčí části také zjednodušuje zobrazení dat v grafech.

Pouze část systémů poskytuje zobrazení dat v grafech. Rozhodl jsem grafy do projektu začlenit, protože zjednodušují pohled na jinak komplexní data.

Primárním účelem systému je dát uživateli možnost vytvořit projekt skládající se z jednotlivých milníků, úkolů a událostí. K takto vytvořenému projektu je možno přiřadit pracovníky. Zakladatel projektu je na začátku vždy pozici vedoucího projektu. Vedoucích může být více a rozlišujeme tedy dva druhy pracovníků projektu - *pracovník* a *vedoucí*.

Milníky označují v systému významnou část práce na projektu, skládající se z úkolů. Tyto se mezi milníky nemohou nijak ovlivňovat. Jednotlivé úkoly v milníku tvoří stromovou strukturu, tudíž jeden úkol může mít více pod-úkolů. Také na sobě mohou záviset tak, že dokud se nesplní jeden úkol, nemohou být splněny úkoly na něm závislé.

Úkoly mohou nabývat několika stavů, přesněji: nový, rozpracovaný, dokončený, nedostupný a nedokončený.

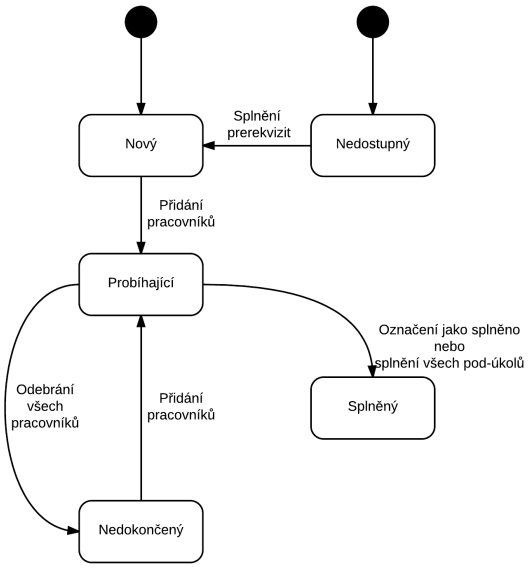
Nový
: Úkol je nově vytvořen, na úkolu zatím nikdo nepracoval. V tomto stavu se také nacházejí nově zpřístupněné úkoly, které byly původně nedostupné, protože jejich prerekvizity nebyly splněny.

Rozpracovaný
: Do tohoto stavu se úkol dostane po přiřazení pracovníků, značí tedy, že na úkolu se pracuje, ale ještě není splněn. Pouze z tohoto stavu může být úkol označen jako splněný.

Splněný
: Dokončený úkol přechází do stavu Splněný, tento stav je konečný a již nejde změnit. V tomto stavu nejsou k úkolu přiřazení žádní pracovníci.

Nedokončený
Tento stav označuje úkol, na kterém již někdo pracoval, ale řešení úkolu vzdal nebo byl odebrán.

Nedostupný
Pokud má úkol při vytváření zadané závislé úlohy, pak je jeho počáteční stav označen jako nedostupný, dokud nejsou všechny tyto úlohy hotovy. Po splnění těchto prerekvizit pak úkol přechází do stavu Nový.



Obrázek 1: Stavový diagram úkolu

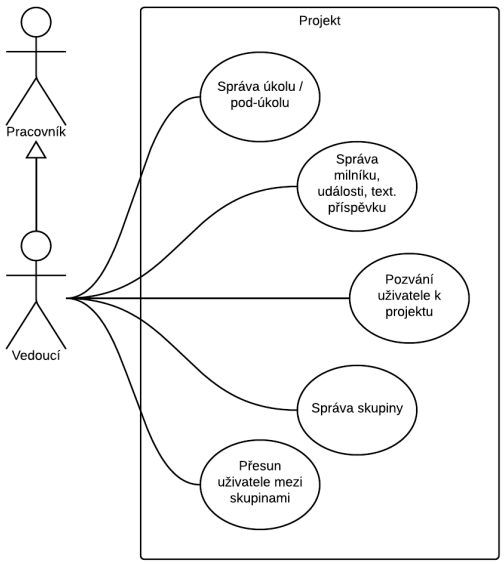
Uživatelé pracující na projektu jsou řazeni do skupin. V projektu figuruje vždy jedna skupina pro vedení projektu a neomezený počet skupin pro ostatní pracovníky. Jeden uživatel nemůže být členem více skupin zároveň ve stejném projektu. Uživatelé nebo i celé skupiny se mohou podílet na řešení úkolů.

K projektu je také možné přidat události a textové příspěvky, které slouží pro komunikaci mezi vedením a týmem. K milníkům, událostem a příspěvkům lze přidávat komentáře pro zpětnou vazbu pracovníků.

Systém bude postaven tak, aby byla možná snadná lokalizace do libovolného jazyka.

4.1 Role

Uživatel se může vzhledem k projektu nacházet v celkově dvou rolích: pracovník a vedoucí. Následující sekce podrobně popíše pravomoci každé role.



Obrázek 2: Use case diagram pro roli Vedoucí

4.1.1 Vedoucí

V této roli se uživatel automaticky ocitne pokud byl osobou jež založila projekt. Vedoucím se stane každá osoba, která se nachází ve skupině pro administrátory projektu, specifikované při jeho založení. Odebráním uživatele ze skupiny jsou mu práva vedoucího odebrána.

Pouze uživatel v roli vedoucího může přidávat k projektu nové pracovníky a měnit skupiny stávajících. Mezi vedoucími již další role nejsou, všichni uživatele v této roli mají stejné pravomoci. Vedoucí mají všechny pravomoci pracovníků.

4.1.2 Pracovník

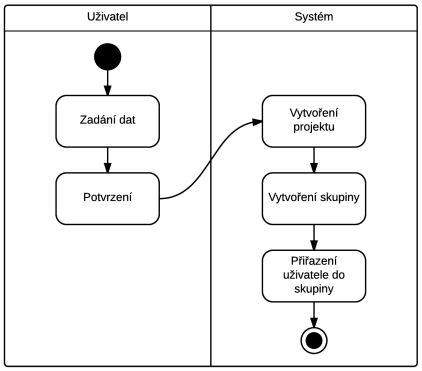
Každý uživatel pracující na projektu, který se nenachází ve skupině s vyššími pravomocemi je v roli pracovník. Tato role umožňuje uživatel zobrazovat data o milnících, úkolech a událostech, přijímat otevřené úkoly, případně se z řešení úkolu odebrat a označit úkol za kompletní. Pracovník nemůže nijak zasahovat do struktury projektu, toto je vše v pravomoci vedoucího.

4.2 Funkční specifikace

Následující sekce budou obsahovat souhrn hlavních netriviálních funkcí systému, které musí obsahovat.

4.2.1 Vytvoření projektu

Tuto akci může provést jakýkoliv přihlášený uživatel. Uživatel musí zadat název daného projektu, jež může doplnit popisem, který detailně zprostředkuje budoucím pracovníkům cíl projektu. Dále musí uživatel zadat datum uzávěrky, tedy požadované datum splnění projektu. Posledním požadavkem na uživatele je zadání názvu výchozí skupiny pro vedoucí projektu. Průběh funkce je znázorněn obrázku 3.



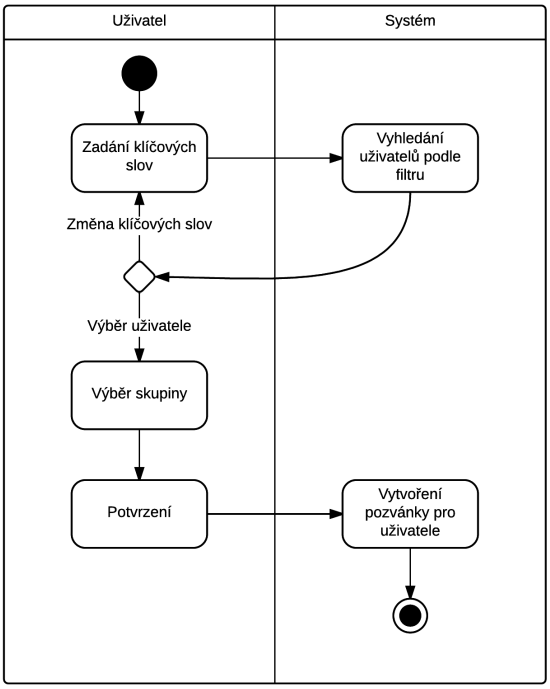
Obrázek 3: Diagram aktivit pro vytvoření projektu

4.2.2 Pozvání uživatele k projektu

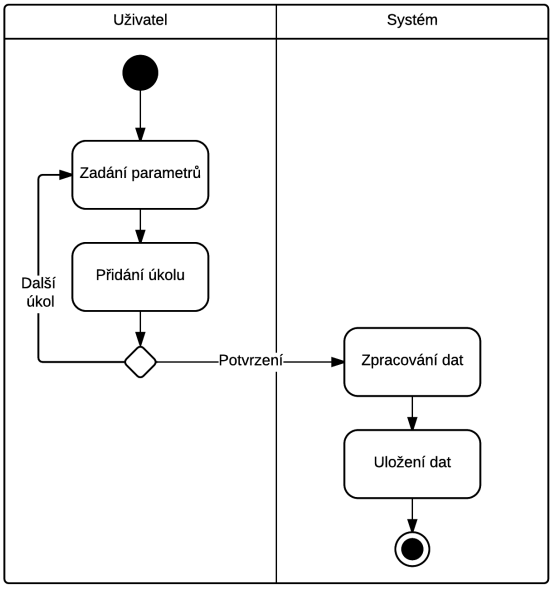
Pokud je vedoucí projektu pracuje s aplikací v online režimu, pak může k projektu pozvat uživatele. Prvním krokem je vyhledání uživatele pomocí klíčového slova — jména, příjmení nebo emailu. Uživatel musí dále ze seznamu vyhledaných uživatelů vybrat požadovanou osobu a následně zvolit skupinu, do které jej pozvat. Potvrzením je uživateli zaslána žádost. Diagram na obrázku 4 znázorňuje průběh této funkce.

4.2.3 Přidání úkolů / pod-úkolů

Uživatel má dvě možnosti přidání úkolů. Hlavní úkoly přiřazované přímo milníku nebo pod-úkoly přiřazované jednotlivým úkolům. Jak je zobrazeno na obrázku 5, po zvolení entity k přidání úkolů je uživatel požádán o určení parametrů úkolů (zda je úkol otevřený či uzavřený, zda obsahuje časovou složku, zvolit závislé úkoly a zadat požadovaný počet pracovníků) a jména. Uživatel může před odesláním požadavku přidat více úkolů.



Obrázek 4: Diagram aktivit pro pozvání uživatele do skupiny



Obrázek 5: Diagram aktivit pro přidání úkolů

5 Návrh systému

V sekci návrhu systému popíši můj návrh serverové a klientské části IS, rozložení jednotlivých vrstev serverové části a strukturu databáze.

5.1 Serverová část

Serverová část systému se skládá z několika částí starajících se o persistenci a čtení dat z úložiště, doménovou logiku a WCF služby pro propagaci dat klientům. Následující sekce budou popisovat každou vrstvu podrobně.

5.1.1 WCF služba

Jak již bylo zmíněno, pro příjem a odesílání dat do a ze serveru je využito rozhraní postavené na technologii WCF. Tato technologie definuje přesně danou množinu funkcí, které klienti mohou využívat. Serverová část poskytuje celkem dvě služby:

Access service je služba pro registraci uživatele, která je dostupná bez nutnosti zadávat jakékoliv přihlašovací udaje či se jinak autorizovat.

Authorization service je přístupná pouze s platnými přihlašovacími údaji, tato služba řeší veškerou komunikaci se serverem po přihlášení. Všechna komunikace probíhá přes šifrovaný protokol HTTPS, pomocí SOAP zpráv s ověřením údajů.

Tato vrstva se stará o kontrolu privilegií při konání akcí, například pokud chce uživatel vytvořit nový milník, tak je zkontrolováno, zda k tomuto úkonu má dostatečná oprávnění. Veškeré výjimky a chybové stavy, které je třeba propagovat klientské části jsou zajištěny pomocí tzv. *FaultContract*. Každá metoda služby je označena anotací *FaultContract* spolu s typem výjimky jež může nastat. Při chybovém stavu, například pokud uživatel nemá dostatečná práva, není možno akci provést nebo nastala jiná chyba pouze vyvolám *FaultException* a vlastní objekt výjimky, který je následně zaslán do klienta k dalšímu zpracování.

V mém systému využívám tyto výjimky pro informování klienta o chybových stavech:

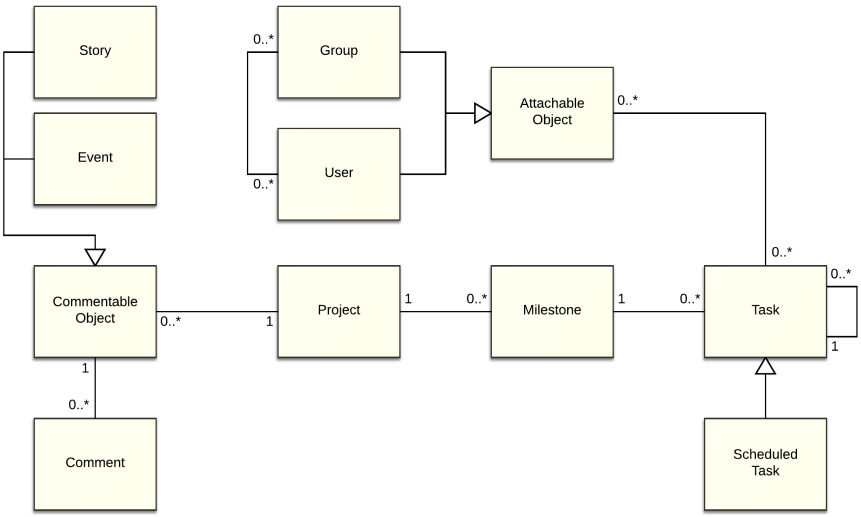
- *fthrEmptyGroupException* – Tato výjimka nastává při pokusu odebrat posledního uživatele ze skupiny pro správce.
- *fthrNotMemberException* – Uživatel nemůže nijak zasahovat do úkolu ke kterým není přiřazen. Tato výjimka nastává pokud se o takovou akci pokusí.
- *fthrPermissionException* – Pokud se uživatel pokusí provést akci, ke které nemá dostatečné pravomoci, bude mu vyvolána tato výjimka.

- *fthrProtectedEntityException* – Tato výjimka nastává při pokusu odebrat chráněný objekt, např. skupinu vedoucích.
- *fthrWorkerLimitException* – Tato vyjímka je vyvolána pokud by přidáním uživatelů byl překročen limit pracovníků projektu.

Důležitou částí této služby je také funkce kontroly přihlašovacích údajů a šifrování hesel. Při registraci uživatele je jím zadané heslo zašifrováno pomocí metod z namespace *System.Security.Cryptography*. Pro kontrolu přijatých přihlašovacích údajů se využívá třída vycházející z abstraktní třídy *UserNamePasswordValidator*. Přihlašovací údaje se kontrolují při každém volání metody WCF služby.

5.1.2 Doménová vrstva

Tato vrstva leží mezi servisní a datovou vrstvou. Obsahuje doménové objekty a výčtové typy, které obě vrstvy využívají. Datová vrstva využívá tyto objekty pro vytvoření a mapování databáze, zatímco WCF služba tyto objekty využívá jako objekty pro přenos dat. Všechny třídy, které využívá servisní vrstva jsou označeny anotací *DataContract* a všechny její properties, které se budou serializovat *DataMember*. Jednotlivé části výčtových typů jsou označeny *EnumMember*.



Obrázek 6: Zjednodušený diagram doménových tříd

5.1.3 Datová vrstva

Vrstva přístupu k datům se stará a uložení a čtení dat z pevného úložiště. Hlavní technologie využívaná v této vrstvě je Entity Framework. Základním kamenem pro EF je tzv.

Context. V mém případě se tento context nazývá *HostContext*. Tento objekt řadu obsahuje *DbSet* kolekcí doménových tříd, které budou namapováný na tabulky databáze a přes které se bude s databází pracovat.

Další část datové vrstvy je sada repozitářu, které pracují přímo s kontextem a slouží k ukládání a vyhledávání objektů v databázi. Každý repozitář v konstruktoru dostane context nad kterým má pracovat. Jelikož systém požaduje, aby se některé úlohy provedly jako transakce, všechny tyto objekty jsou sjednoceny v návrhový vzor *Unit Of Work*. Entity framework dokáže sám o sobě sledovat jednotlivé entity a ví tedy, zda jsou nové, upravené či smazané. Tato funkce je využita pro implementaci transakcí v datové vrstvě. Při provádění jedné úlohy je vytvořena pouze jedna instance třídy HostContext, která se sdílí mezi všemi repozitáři. To má na starost třída jež jsem nazval *UnitOfWork*. Tato třída je schopna získat instanci jakéhokoliv repozitáře a vrátit ji již s nastaveným contextem. Pro provedení transakce po provedení všech změn stačí zavolat pouze funkce *Save*.

Poslední část tvoří automatické migrace databáze. Tato funkce je součástí Entity Framework. Každá změna modelu je zaznamenána a převedena do kódu. Pomocí dvou jednoduchých příkazů tak umožňuje jednoduchý návrat k předchozím verzím databáze.

5.2 Klientská část

Klientská část systému se skládá hlavně z vrstvy pro komunikaci se serverovou službou, aplikační vrstvy starající se o výpočty a logiku, prezentační vrstvy pro vstup a výstup programu a speciální části starající se o běh aplikace na pozadí.

5.2.1 Service consumer

Projekt je nastaven pro odběr obou služeb ze serveru přidáním reference služby. WCF si automaticky vytvoří vlastní namespace se třídami definovanými jako *DataContract* a třídou *Client* reprezentující všechny metody dané služby. Do vlastnosti *Credentials* instance této třídy se vkládají přihlašovací údaje uživatele, které se mají na serveru porovnat. Client poskytuje synchronní i asynchronní verze metod, asynchronnost je zajišťována pomocí *Event-based Asynchronous Pattern* (EAP). Pro WP jsou generovány pouze asynchronní metody.

V této vrstvě jsou také odchytávány výjimky vyvolané pomocí *FaultException*.

5.2.2 Logická vrstva

Logická vrstva se stará o zpracování dat při komunikaci mezi prezentační a servisní částí klienta. Tato část systému tedy přijme zadaná data, provede nejzákladnější validace (prázdná pole, startovní datum větší než datum konce, ...) a předá data ke zpracování. Další funkcí této vrstvy je zpřístupnění offline provozu. Nejdůležitější přijatá data jsou

ukládána do vnitřní databáze telefonu, jak bude popsáno v další sekci. Stejně tak při získávání dat dokáže tato vrstva rozlišit, zda je aplikace v offline nebo online módu a získá tedy data z tomu určeného zdroje.

Jak již bylo zmíněno, aplikace dokáže pracovat ve dvou módech

5.2.2.1 Online mód – Základní způsob práce s aplikací. Po úspěšném kontaktu se serverovou službou je uživatel přihlášen a aplikace pracuje v online módu. Tento mód poskytuje veškeré funkce systému. Pokud je zařízení nepřipojeno k internetu, pokud síla signálu není dost velká nebo pokud se nepodařilo navázat spojení se serverem, pak je mód aplikace přepnut na *offline*.

5.2.2.2 Offline mód – V tomto módu telefon nevyžaduje připojení k serverové části a všechna data čte ze své vnitřní databáze, kde byla data dříve uložena z online modu. Zasílání dat na server je omezeno jen na některé funkce, například vyhledání uživatele pro účel pozvání k projektu není možno realizovat. Tyto akce, které pro svou funkčnost přímo nepotřebují připojení k internetu jsou uloženy k pozdějšímu odeslání.

5.2.3 Databáze a datová vrstva

Pro persistenci dat je použit návrhový vzor *Data mapper*. Přes sadu těchto mapperů jsou ukládány jednotlivé objekty do databáze pro pozdější užití.

5.2.4 Background Task Agent

Background Task Agent má na starosti periodické spouštění kódu na pozadí, tzn, když aplikace není používána. Tato část systému má na starosti celkem dvě funkce: periodickou kontrolu připojení a následné odeslání odložených akcí a získávání uživatelských upozornění ze serveru.

- *Task Queue* – Tato funkce pravidelně kontroluje, zda lze dosáhnout komunikace s serverem a pokud ano, pak se pokusí postupně odeslat všechny zprávy, které byly uloženy v offline módu. Nezdařené dotazy jsou označeny jako *failed* a již se znovu neodesílají. Tyto dotazy jsou předloženy uživateli k opětovnému odeslání či smazání později.
- *Notification manager* – Notification manager pravidelně odesílá na server požadavek pro získání upozornění ohledně daného uživatele.

Tento agent je spouštěn každých 20-40 minut, podle vytížení systému, na maximálně 25 sekund. Pokud je telefon v úsporném režimu, tento kód se vůbec nespouští. Uživatel má možnost vypnout jednotlivé funkce BTA.

5.2.5 Prezentační vrstva

Hlavní částí klienta je prezentační vrstva, která se stará o komunikaci s uživatelem. Pro zobrazení obsahu WP využívá technologii XAML (detailněji popsána v sekci Použité technologie 3). Ve stránkách se neprovádí žádné logické operace, pouze se předávají data do dalších vrstev systému.

6 Implementace serverové části

V této části se budu zabývat tématem implementace serverové části systému. Jak již bylo zmíněno v předchozích odstavcích, tato část se skládá ze tří základních částí: části pro uložení dat, části webové služby a části obsahující doménové třídy. Na rozdíl od sekce 5 se bude tato sekce zabývat samotným řešením a programováním jednotlivých celků.

6.1 WCF služby

Tato vrstva obsahuje celkem dvě služby pro komunikaci, jedna poskytující spojení bez nutnosti autentifikace, určené pro akce, kdy uživatel nemá vlastní přihlašovací údaje a jedna pro komunikaci ověřovanou pomocí uživatelského jména a hesla. Tohoto je docíleno pomocí nastavení v konfiguračním souboru služby.

```
<!--Access service-->
<binding name="basicHttpBindingConf">
    <security mode="Transport">
        <transport clientCredentialType="None" />
    </security>
</binding>
<!--Auth service-->
<binding name="basicHttpAuthBindingConf">
    <security mode="TransportWithMessageCredential">
        <transport clientCredentialType="Basic" />
    </security>
</binding>
```

Výpis 1: Nastavení *bindings* v App.Config

Proto aby služby s klientskými aplikacemi bezpečně komunikovaly protokolem *https* bylo toto chování také nutno nastavit v konfiguračním souboru *App.Config*. Celkem jsou třeba změny dvou nastavení. První z nich je nastavení koncového bodu (endpoint), přesněji nastavení hodnoty *binding* na *mexHttpsBinding* u koncového bodu pro výměnu Metadat a parametr *base adress* na hodnotu IP adresy serveru pro dosažení služby s protokolem *https*. Další potřebnou změnou je nastavení SSL certifikátu. Pro účely této práce je využit tzv. self-signed certifikát, tedy vlastně vytvořený certifikát určený pro testování. Nastavení certifikátu se určuje v sekci *behavior*.

```
<behavior name="AccessServiceBehavior">
  <serviceCredentials>
    <serviceCertificate findValue="PCName" storeLocation="LocalMachine"
            storeName="Root"
            x509FindType="FindBySubjectName" />
  </serviceCredentials>
</behavior>
```

Výpis 2: Nastavení *behavior* v App.Config

Nastavením těchto hodnot je služba připravena přijímat a odesílat data po zabezpečeném kanálu. Jak již bylo zmíněno, jedna ze služeb je nastavena tak, aby se při každém dotazu na službu klient ověřoval uživatelským jménem a heslem. O validaci těchto přihlašovacích údajů se stará třída *UserValidator*. Tato třída dědí ze třídy *UserNamePasswordValidator* a přepisuje jeho metodu *Validate*. V této metodě jsou data porovnána se zakódovaným heslem odpovídající přijatému uživatelskému jménu. V případě neshody je pomocí *FaultException* oznámen chybový stav. Tento validátor je přiřazen ke autentifikační službě v *App.Config*.

```
<serviceCredentials>
  <userNameAuthentication userNamePasswordValidationMode="Custom"
        customUserNamePasswordValidatorType="DBAccessServiceHost.Service.
            UserValidator,DBAccessServiceHost" />
        ...
</serviceCredentials>
```

Výpis 3: Registrace validátoru v App.Config

Samotné rozhraní služeb lze najít v samostatných souborech *IAccessService.cs* a *IAuthService.cs*, které obsahují předpisy všech metod, která služba poskytuje. Třídy *AccessService* a *AuthService* tato rozhraní implementují. V těchto třídách se nachází veškerá logika WCF služby. V metodách se využívá objekt *UnitOfWork*, tento objekt bude detailněji popsán v dalších sekcích.

6.2 Doménové třídy a výčtové typy

V systému je celkem šestnáct doménových tříd a dva výčtové typy. Tyto třídy jsou mapovány do databáze a v repozitářích jsou přímo užívány k ukládání dat. Každá doménová třída, která se mapuje do databáze dědí ze třídy *BaseEntity*, která rozšiřuje objekty o položky *DateModified* a *DateCreated*. Každá časová položka je automaticky upravována díky přetížení funkce *SaveChanges* v Contextu systému. Tato funkce bude detailněji popsána v dalších sekcích.

6.2.1 Doménové třídy

Ze všech šestnácti doménových objektu, ne všechny jsou přímo ukládány jako vlastní tabulky do databáze. Mezi objekty existuje dědičnost, která se do pevného úložiště mapuje podle vzoru *Single table inheritace*. Přesněji jsou to třídy *CommentableObject*, *AttachableObject* a *TaskComponent*.

AttachableObject označuje třídu, jejíž instanci je možno přiřadit k úkolu. Všechny třídy, které ji dědí jsou tedy jednotlivci nebo uskupení osob, kteří se na řešení úkolu mohou podílet.

Třídy, které dědí *AttachableObject*: *User*, *Group*.

CommentableObject je třída, ke které lze přidávat komentáře. Objekt této třídy musí náležet k některému projektu.

Třídy, které dědí *CommentableObject*: *Story*, *ProjectEvent*.

TaskComponent je třída samotného úkolu. K objektům této třídy je již možno přiřazovat skupiny i uživatele.

Třída, která dědí *TaskComponent*: *ScheduledTask*.

6.2.2 Výčtové typy

Dva výčtové typy využívané těmito třídami jsou *State* a *Color*. Výčet *State* obsahuje označení všech stavů, ve kterých se může nacházet objekt *TaskComponent*. Hodnoty toho výčtu přímo korespondují se stavy zmíněné v sekci 4.

Výčtový typ *Color* obsahuje všechna možná barevná označení, která jdou k entitám (milníky, úkoly, události) přiřadit. V systému je počet barev omezen na vyčleněných 20-ti barev, které jsou dobře viditelné na světlém i tmavém pozadí. Pokud by uživatel vybíral toto barevné označení sám, mohl by tím ovlivnit ostatní uživatele, kteří na svých klientech nepoužívají stejný barevný motiv. Názvy barev přímo korespondují s barevnými motivy Windows Phone.

6.3 Unit of Work

Pro implementaci transakcí je využit návrhový vzor Unit of Work, který zaobaluje všechny repozitáře. Entity Framework sám sleduje změny v entitách, které se v databázi nachází. Této funkce jsem využil přo řešení této problematiky. EF rozlišuje tyto stavy entit:

- *Added* – Entita je v contextu nová a je označena pro přidání do databáze
- *Modified* – Entita již v contextu existuje a byla upravena. Po uložení změn bude upravena.
- *Unchanged* – Entita již v contextu existuje, ale nebyly probedeny žádné změny.
- *Deleted* – Entita byla označena pro smazání, po uložení změn bude odebrána ze systému.
- *Detached* – Entita není v contextu.

Klíčové pro tyto stavy entit je akce, která se provede až po zavolání metody *SaveChanges*. Tato vlastnost je pro implementaci klíčová.

Repozitáře se starají o změny doménových objektů nad contextem. V systému existuje jeden generický repozitář *GenericMapper*, který každý repozitář rozšiřuje o speciální případy vlastních metod. Každý takový repozitář musí obsahovat odkaz na context, se kterým pracuje a sadu metod, které podporuje. Těmito metodami jsou:

- *Find* – Vyhledání podle ID entity
- *Update* – Úprava entity
- *Delete* – Smazání entity
- *Insert* – Vložení nové entity do contextu.

Tyto metody slouží pouze ke změně stavu entity (EntityState) a voláním těchto metod zatím není provedena žádná fyzická změna v datech. Context, na něž má každý repozitář referenci a nad nímž pracují, je sdílen mezi všemi těmito objekty. Až samotným zavoláním metody *Save()* v Unit of Work jsou všechny tyto entity předány ke zpracování.

6.4 HostContext

Zde bych se chtěl zaměřit jen na metodu ze třídy DBContext a tou je *SaveChanges()*. Metoda *SaveChanges()* slouží pro potvrzení změn v databázi. Přepsáním této metody jsem docílil automatického zápisu polí *Created* a *Modified* u tříd dědících *BaseEntity*. Tato metoda projde všechny změněné entity a u nově přidaných a modifikovaných objektů změní příslušné datum.

```
public override int SaveChanges()
{
    var entities = ChangeTracker.Entries()
    .Where(x => x.Entity is BaseEntity && (x.State == EntityState.Added || x.State ==
        EntityState.Modified));
    foreach (var entity in entities)
    {
        if (entity.State == EntityState.Added)
        {
            ((BaseEntity)entity.Entity).Created = DateTime.Now;
        }

        ((BaseEntity)entity.Entity).Modified = DateTime.Now;
    }

    return base.SaveChanges();
}
```

Výpis 4: Override metody SaveChanges

7 Implementace WP části

Klientská část projektu je mobilní aplikace pro Windows Phone. Zde se zastavím hlavně nad implementací metody konzumace WCF služby, uložení uživatelských dat a offline provozu, lokalizace programu a popíšu uživatelské rozhraní.

7.1 AuthDataResolver, AuthDataSender

Pro příjem a odesílání dat do sítě jsou používány třídy *AuthDataSender* a *AuthDataResolver*. První řečená třída se stará o samotné odesílání a příjem dat, všechny metody jsou asynchronní. Druhá třída zaobaluje metody třídy první a obsluhuje zachytávání výjimek, výběr správného datového úložiště a předává data k uložení pro offline provoz.

7.1.1 Data Sender

Třída data sender k vytvoření instance poskytuje dva konstruktory. Implicitní konstruktor, který si sám získá přihlašovací data ze dříve uložených dat a parametrický konstruktor, jemuž jsou dodány přihlašovací údaje jako parametry. Tento konstruktor také otevře komunikační kanál. Třída implementuje rozhraní *IDisposable* v jehož metodě *Dispose* je tento kanál uzavřen.

Volání asynchronních metod WCF služby i získání výsledku je řešeno v rámci jedné metody. Toho je docíleno vytvořením lambda funkce pro událost *_Completed*. V této funkci se přijme a vrátí výsledek pomocí *TaskCompletionSource*, případně se nastaví a vrátí výjimky.

```
public Task<Project> GetProjectData(int id)
{
    var tcs = new TaskCompletionSource<Project>();

    proxy.GetProjectCompleted += (sender, e) =>
    {
        if (e.Error != null)
            tcs.TrySetException(new MyDataException(AppResources.ErrorTitle,
                AppResources.ErrorGettingData));
        else
            tcs.SetResult(e.Result);
    };

    proxy.GetProjectAsync(id);

    return tcs.Task;
}
```

Výpis 5: Asynchronní metoda pro získání dat o projektu

7.1.2 Data Resolver

Třída *AuthDataResolver* má tři základní funkce: rozhodnutí, zda se z daty bude pracovat v offline či online režimu a následný způsob jejich uložení či nahrání, ukládání přijatých dat do vlastní databáze telefonu a zachytávání výjimek. První funkce je zajištěna globální proměnnou *OfflineMode* jež je nastavována při startu programu a případně upravena při pokusech o připojení k serveru. Kontrolou stavu této proměnné je tedy rozhodnuto v jakém módu telefon pracuje.

V online modu tato třída přímo spolupracuje s *AuthDataSender* pro příjem a zasílání dat. Pokud metoda přijímá data ze serveru, pak jsou tato data pomocí *mapperů* uloženy do interní databáze. Pokud se program nachází v offline modu, pak je pro získání dat použita právě databáze telefonu, tato databáze obsahuje pouze dříve získaná data. Naopak, pro odesílání je volání odloženo do fronty příkazů, jež se spustí, jakmile telefon přejde do stavu online. Tato fronta je realizována vlastní databázovou tabulkou, která obsahuje akci, jež se má provést a parametry, se kterými se má odeslat. Pro snadné uložení dat jsou všechny parametry uloženy ve formátu JSON, k čemuž dopomáhá knihovna Newtonsoft Json.

Pro zpracování výjimek, které propagují chybovou hlášku uživateli je vytvořena vlastní výjimka *MyDataException*, která obsahuje metodu *ShowPrompt()*.

Obrázek 7: Příklad chybové hlášky

Pro zobrazení hlášky je využita komponenta z knihovny *Coding4Fun.Toolkit*, která umožnuje zobrazit oznámení podobné nativním notifikacím ve WP.

7.2 Uložení uživatelských dat

Po prvním uživatelově úspěšném přihlášení jsou jeho údaje bezpečně uloženy do paměti telefonu jednak pro pozdější užití při provádění akcí a jednak pro automatické přihlášení při příštím spuštění aplikace. Pro uložení těchto dat jsem zvolil tzv. *Credential Locker*, který dokáže uložit data na bezpečné úložiště a ukládaná data zašifruje. Další užitečnou funkcí *Credential Lockeru* je možnost migrace uložených dat mezi zařízeními se stejným Microsoft účtem.

7.3 Lokalizace

Pro lokalizaci aplikace je využit standardní přístup překladu pomocí *Resource* souborů, tedy souborů obsahující přeložené textové řetězce přiřazené k určitému klíči. V době

psaní této práce obsahuje program dva soubory *.resx*, výchozí soubor obsahující anglické texty a soubor s českou lokalizací programu. K těmto řetězcům se pak jednoduše přistupuje přes třídu *AppResources*. V klientu jsou přeloženy všechny chybové hlášky i uživatelské rozhraní.

8 Uživatelské rozhraní

V této části práce bych se chtěl věnovat nejdůležitější části mobilního klienta a tím je uživatelské rozhraní. Pokusím se popsat a vysvětlit funkci vlastních definovaných komponent, které jsem v práci využil. Dále se zaměřím na prvky externích komponent a mé řešení vizualizace dat v grafech. Poslední částí bude seznámení s programem jako takovým.

8.1 Prvky specifické pro Windows Phone

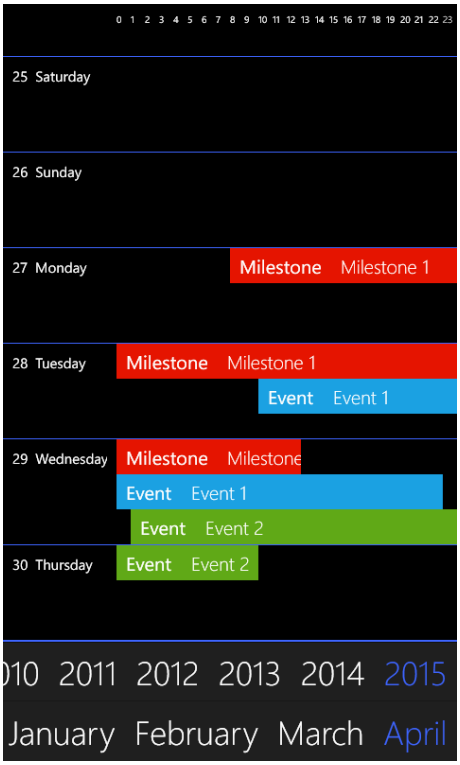
Uživatelské rozhraní Windows Phone 8 obsahuje všechny základní komponenty jako jeho konkurenti Android a iOS. Od těchto platforem se ovšem odlišuje svým typickým způsobem navigace v aplikacích pomocí prvků *Panorama* a *Pivot*. I když jsou tyto komponenty na první pohled velice podobné — až identické, ve způsobu jejich použití je velký rozdíl. *Panorama* je používáno hlavně pro hlavně pro hlavní obrazovky za účelem navigace k dílčím částem aplikace. Jak název *Panorama* napovídá, obsah stránky v této komponentě zasahuje i mimo viditelnou část obrazovky. Pomocí zabudované dotykové funkce je možno jednotlivé části této komponenty procházet. *Pivot* použijeme tam, kde bychom na jiných platformách použili *Tab control* (záložky), jeho případem užití je tedy zobrazení samotných dat. Stejně jako *Panorama*, *Pivot* také poskytuje dotykové ovládání a přechod mezi stránkami. Moje aplikace zobrazuje čistě data a jakýkoliv „hub“ v ní není potřeba, pro navigaci tedy užívám hlavně komponentu *Pivot*.

8.2 Vlastní komponenty

V projektu jsem definoval celkem dvě vlastní komponenty, které pomáhají zobrazit data, pro které není na platformu Window Phone určený prvek. Těmito komponentami jsou *Calendar* a *TreeViewControlItem*.

Pro zobrazení kalendáře existují externí knihovny, ty ovšem nesplňovaly potřeby této aplikace a proto jsem definoval vlastní *UserControl*, který dokáže zobrazit nejen dny v týdnu a měsíce v roce, ale do jednotlivých dnů dokáže zaznamenat jednotlivé milníky, události a úkoly projektu. Čas začátku a konce konání jsou určeny horizontálním posunutím prvku kalendáře. Kliknutím na událost kalendáře se zobrazí její detail.

Jelikož v systému hraje velkou roli stromová struktura úkolů s pod-úkoly, bylo zapotřebí tato data nějakým způsobem zobrazit. Pro řešení jsem vyzkoušel celkem tři metody z nich následující jsem shledal nejvíce vhodnou. První části řešení je *UserControl TreeViewControlItem*, který zobrazuje veškerá data, například název, stav a počet pod-úkolů, úkolu do jedné komponenty. Všechny úkoly první úrovně (bez nadřazeného úkolu) jsou následně zobrazeny v *Pivot* komponentě. Zvolením jednoho úkolu je do *Pivotu* přidána stránka do než jsou vypsány všechny pod-úkoly vybraného úkolu. Stejným principem se pracuje s nižšími úrovněmi stromové struktury.



Obrázek 8: UserControl Calendar

8.3 Converter

Často je při tvorbě uživatelského rozhraní potřeba zobrazit data, která ale v základní podobě nemají správnou formu. Je třeba je tedy modifikovat a k tomu využívám třídy implementující rozhraní *IValueConverter*. Toto rozhraní předepisuje metody *Convert* a *ConvertBack* pro konverzi objektů na jiný a zpátky. *Converter* je využíván hlavně při vázání dat ke komponentám UI v XAML. V mé aplikaci využívám tyto „konvertéry“:

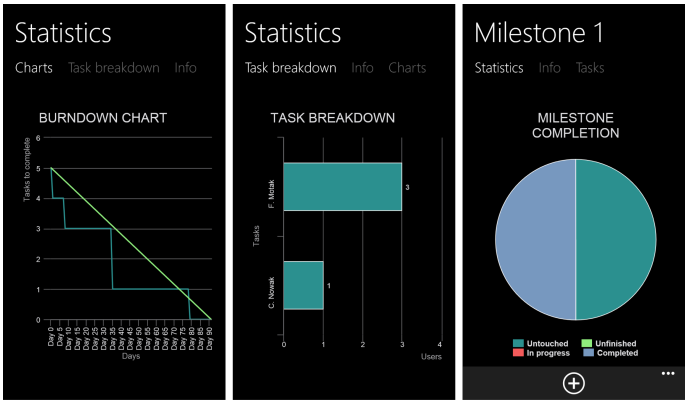
- *ActionToTextConverter* – převádí hodnoty číselníku *fthrAction* na lokalizované textové hodnoty;
- *BooleanToVisibilityConverter* – převádí booleovskou hodnotu na danou hodnotu viditelnosti prvku. Tento converter se používá často pro skrytí a zobrazení grafických prvků v závislosti na proměnné;
- *InvertedBooleanToVisibilityConverter* – stejná funkce jako předchozí třída, ale s invertovanou funkcí;
- *NewsObjectToTextConverter* – převádí objekt na text podle jeho typu pro použití v novinkách projektu;

- *ObjectToColorConverter* – pro převod hodnoty z výčtu *fthrColor* na samotnou barvu objektu;
- *ObjectToImageConverter* – vrací obrázek asociovaný k typu objektu;
- *ObjectToNameConverter* – podobná funkce jako *NewsObjectToTextConverter*, vrací jiné textové hodnoty;
- *StateToImageConverter* – převádí stav úkolu na obrázkovou variantu;
- *StateToTextConverter* – převádí stav úkolu na přeloženou textovou hodnotu;
- *ZeroToInfinityConverter* – převede znak „0“ na znak „∞“.

8.4 Grafy

Původním záměrem bylo použít pro zobrazení grafů kombinaci knihoven *amCharts* a *Sparrow Toolkit*. Knihovna *amCharts* poskytuje možnost zobrazení koláčových grafů (Pie chart), ostatní druhy grafů ovšem nejsou plně podporovány. Nedostatky této knihovny měl doplnit *Sparrow Toolkit*, který obsahuje grafy chybějící v předešlé knihovně. Tyto balíky byly nicméně graficky nekonzistentní a obsahovaly spousty drobných chyb, kvůli kterým jsem se rozhodl od těchto knihoven upustit.

Jako alternativu pro grafy jsem nakonec zvolil *JavaScriptovou* knihovnu *HighCharts*. Jelikož se nejedná o knihovnu postavenou na technologii .NET, ale je určena pro zobrazení ve webovém prohlížeči, všechny grafy využívají komponentu *WebBrowser*. V této komponentě se zobrazí výchozí stránka *chart.html*. Tato stránka sama o sobě neobsahuje žádný z grafů *HighCharts*, teprve vyvoláním metody *highcharts()*. Parametrem této funkce jsou informace o typu, osách, popisech a datech grafu ve formátu JSON. Příklad použitých grafů lze nalézt na obrázku 9.



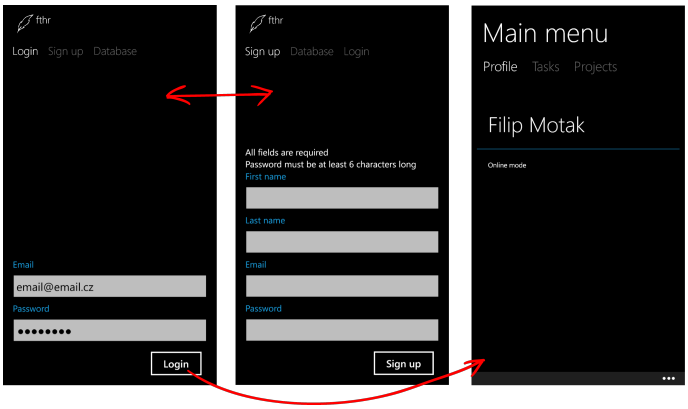
Obrázek 9: Příklad grafů použitých v systému

9 Použití aplikace

V poslední sekci této práce bych se rád věnoval stručnému popisu práce s aplikací. Popíši nejdůležitější úkony, které aplikace poskytuje a nastíním ovládání uživatelského rozhraní. Popis bude proveden z pohledu vlastníka projektu.

9.1 Přihlášení / registrace uživatele

První obrazovkou, která se uživateli zobrazí při novém spuštění aplikace je obrazovka pro přihlášení. Tato stránka se skládá z části pro přihlášení uživatele a z části pro jeho registraci. Mezi stránkami se dá přecházet tahem prstu do levé či pravé strany nebo dotekem položky v horní části obrazovky. Zadáním požadovaných údajů a stisknutím potvrzovacího tlačítka se přihlášení / registrace provede. Navigace stránky je znázorněna na obrázku 10.



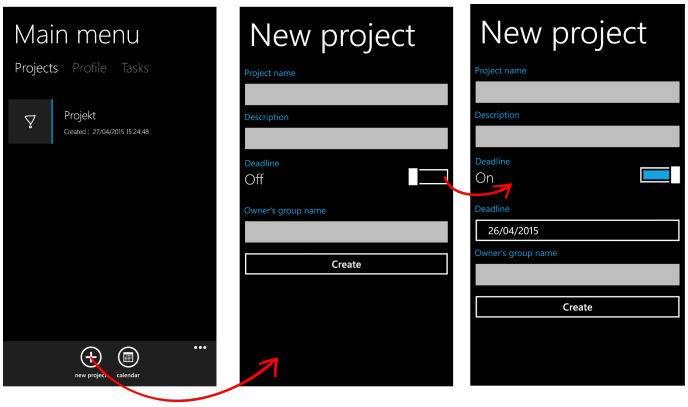
Obrázek 10: Přihlašovací obrazovka

9.2 Uživatelský profil a vytvoření projektu

Přihlášením do systému se uživatel dostane na stránku jež nazývám „hub“. Na této stránce najde nejdůležitější informace o uživateli, seznam úkolů, na kterých pracuje napříč všemi projekty a seznam projektů, jichž je členem. Ze spodního menu je uživateli umožněno se odhlásit, zobrazit nastavení aplikace a zobrazit svůj kalendář. Pokud je se uživatel nachází v sekci s projekty, v menu je navíc možnost pro vytvoření nového projektu.

Zvolením této položky se uživatel dostane na stránku pro tvorbu projektu. Na této stránce se kromě vstupů pro základní data nachází tlačítko pro deadline projektu, který je nepovinnou položkou. Přepnutím tohoto přepínače se zobrazí komponenta pro výběr data. Zadáním všech povinných údajů a potvrzením se založí projekt. Vytvoření projektu je znázorněno na obrázku 11.

Zvolením projektu v seznamu se zobrazí jeho detail.

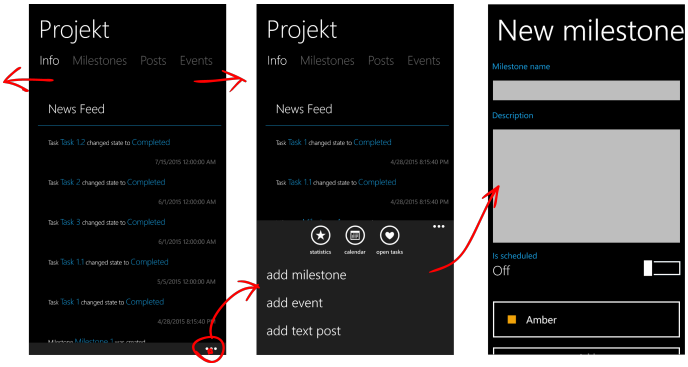


Obrázek 11: Vytvoření projektu

9.3 Detail projektu, založení milníku, události a příspěvku

Detail projektu obsahuje celkem 5 sekcí — novinky projektu, milníky, události, textové příspěvky a seznam pracovníků. Spodní menu obsahuje řadu funkcí pro ovládání projektu a navigaci v projektu. Zvolením příslušných ikon se uživatel může dostat na detail statistik projektu, kalendář a otevřené úkoly. Zvolením položek menu je vlastníku umožněno založit milník, událost a textový příspěvek. Navigace stránky je znázorněna na obrázku 12.

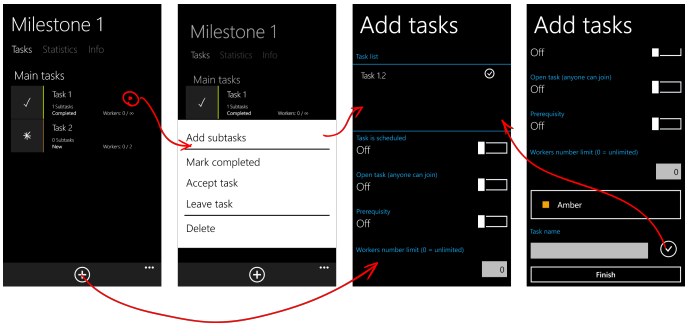
Založení milníku je téměř identické se založením projektu co se týče zadávání dat. Položkou navíc je možnost vybrat barvu, která se u tohoto milníku bude zobrazovat v kalendáři a ostatních seznamech. Potvrzením se zašlou data a daný milník by se měl objevit v seznamu.



Obrázek 12: Vytvoření milníku

9.4 Přidání úkolu / pod-úkolu

Přidání úkolů se volí z obrazovky detailu milníku (viz obr. 13). Úkoly uživatel přidává stisknutím tlačítka ve spodním menu, kdežto pod-úkoly se přidávají podržením vybraného nadřazeného úkolu a vybráním položky z menu. V tomto menu lze také zvolit akce pro řešení úkolu, pokud je úkol otevřený nebo na něm uživatel pracuje.

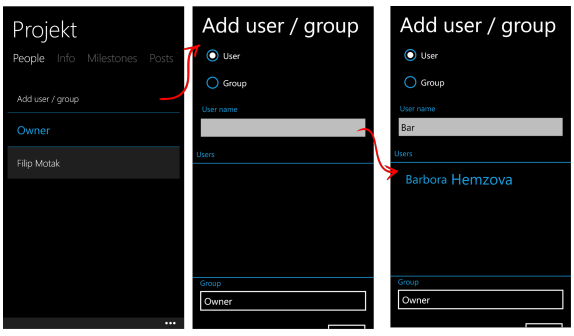


Obrázek 13: Přidání úkolů

Stránky pro přidání úkolů a pod-úkolů jsou identické. Uživatel může přidat více úkolů najednou, u každého úkolu určuje uživatel povinné a nepovinné parametry, barvu a název. Přidané úkoly jsou zobrazeny na vrchní části stránky, z tohoto seznamu je lze odebrat. Stisknutím tlačítka pro dokončení jsou všechny úkoly odeslány.

9.5 Pozvání uživatele k projektu

Zaslání pozvánky k projektu lze provést ze stránky detailu projektu v sekci *Lidé*. Stisknutím tlačítka pro přidání pracovníka se uživatel dostane na stránku pro pozvánky. Tato stránka disponuje možností přidat uživatele a novou skupinu. Požadováného uživatele lze najít pomocí vyhledávacího pole. Všechny shodné výsledky jsou zobrazeny v seznamu. Uživatel poté zvolí vybraného uživatele a skupinu do něž chce uživatele pozvat. Navigace stránky je znázorněna na obrázku 14.



Obrázek 14: Pozvání uživatele

10 Závěr

Cílem práce bylo vytvořit informační systém, který mohou malé společnosti a spolky použít při řešení projektů. Systém nyní umožňuje uživateli vytvářet projekty, dělit jej na jednotlivé části — milníky, úkoly s pod-úkoly a události. Uživateli je umožněno rozdělit pracovníky na dílčí části a sledovat jejich plnění.

V práci jsem popsal řešení tvorby práce od specifikace požadavků, přes analýzu a návrh. Nakonec byla popsána samotná implementace nejdůležitějších a netriviálních částí systému a grafické rozhraní klientské aplikace.

Díky této práci jsem se seznámil s technologiemi, které jsem doposud znal jen okrajově, například Entity Framework a Windows Community Foundation. Nejzajímavější pro mne byl vývoj aplikace pro platformu Windows Phone, zejména tvorba uživatelského rozhraní.

11 Reference

[1] Highcharts AS. Highcharts js, April 2015. URL http://www.highcharts.com.

[2] Basecamp. Basecamp, April 2015. URL https://basecamp.com.

[3] Microsoft MSDN Community. Transport security with basic authentication, April 2015. URL https://msdn.microsoft.com/cs-cz/library/ms733775(v=vs.110).aspx.

[4] Zoho Corporation. Zoho projects, April 2015. URL https://www.zoho.com/projects/.

[5] Tom Dykstra. Implementing the repository and unit of work patterns, July 2013. URL http://www.asp.net/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application.

[6] Martin FOWLER. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003. ISBN 0-321-12742-0.

[7] Huddle. Huddle, April 2015. URL http://www.huddle.com.

[8] Juval LOWY. *Programming WCF Services*. O'Reilly Media, 2010. ISBN 0-596-80548-9.

[9] Microsoft. Windows phone toolkit, April 2015. URL https://phone.codeplex.com.

[10] Newtonsoft. Json.net, April 2015. URL http://www.newtonsoft.com/json.

[11] Defuse Security. Salted password hashing - doing it right, August 2014. URL https://crackstation.net/hashing-security.htm.

[12] Michael Sliwinski. Nozbe, April 2015. URL https://nozbe.com.

[13] Element Studios. Copper, April 2015. URL http://www.copperproject.com.